

Scheduling algorithm to select k optimal programme slots in television channels: A graph theoretic approach

Madhumangal Pal* and Anita Pal†

*Department of Applied Mathematics with Oceanology and Computer Programming,
Vidyasagar University, Midnapore – 721 102, India.
e-mail: mmpalvu@gmail.com; Mobile no. (+91) 9932218937

†Department of Mathematics
National Institute of Technology Durgapur
Mahatma Gandhi Avenue, Durgapur-713 209
West Bengal, India
email: anita.buie@gmail.com

Abstract

In this paper, it is shown that all programmes of all television channels can be modelled as an interval graph. The programme slots are taken as the vertices of the graph and if the time duration of two programme slots have non-empty intersection, the corresponding vertices are considered to be connected by an edge. The number of viewers of a programme is taken as the weight of the vertex. A set of programmes that are mutually exclusive in respect of time scheduling is called a session. We assume that a company sets the objective of selecting the popular programmes in k parallel sessions among different channels so as to make its commercial advertisement reach the maximum number of viewers, that is, a company selects k suitable programme slots simultaneously for advertisement. The aim of the paper is, therefore, to help the companies to select the programme slots, which are mutually exclusive with respect to the time schedule of telecasting time, in such a way that the total number of viewers of the selected programme in k parallel slots rises to the optimum level. It is shown that the solution of this problem is obtained by solving the maximum weight k -colouring problem on an interval graph. An algorithm is designed to solve this just-in-time optimization problem using $O(kMn^2)$ time, where n and M represent the total number of programmes of all channels and the upper bound of the viewers of all programmes of all channels respectively. The problem considered in this paper is a daily life problem which is modeled by k -colouring problem on interval graph.

Keywords: Graph theory, Modelling, design and analysis of algorithms, graph colouring, interval graphs.

2000 AMS Subject Classifications: Primary codes: 05C15; 05C17; 05C85; 05C90; Secondary

codes: 68R10; 68W05

1 Introduction

Today television has acquired the central position of all our means of entertainment. Television is not only the most popular technological device of entertainment but also the best media for sending information in the simplest way. Various production units and advertisement agencies are connected with television. In recent years, most of the world wide mass is being influenced by advertisement. The daily requirements of man is now being governed by the attractive advertisements. The production companies are like to promote their products through different television channels taking the help of advertising companies (ACs) (the companies those are interested to advertise for their parties products). The production companies are spending a large amount of money for the purpose. The ACs are adopting attractive ideas to catch the viewers' attention. The main objective of the ACs are to catch maximum viewers so that the sale of the corresponding product becomes maximum.

There are hundreds of channels like CNN, HBO, STAR, ZEE, MGM, NATIONAL GEOGRAPHIC, AXN, etc., running numerous programmes for 24 hours. Among all these programmes there are some which are very popular. The programmes like 'Guinness World Record Prime Time' at 9:00 a.m. shown at AXN, 'Charlie Chaplin' at 9:00 a.m. on ZEE ENGLISH, 'Mission Everest' at 10:00 a.m. on National Geographic channel and many others are found very much popular. Now the problem arises how and where an AC relays its advertisement to make it viewed by large mass of population. Sometimes, depending on the popularity of the programmes, viewers get divided. Like, in AXN a serial named 'Bay Watch' shown at 8:30 p.m. is very popular and at the same time the programme 'Cindrella' on CNN is also very popular. So, there arises a problem of selecting the channel. If the AC is interested in both the slots it is not be in maximum profit as the viewers get divided.

Nowadays, most of the channels run for 24 hours a day. Suppose BBC has programmes in the respective schedules such as during 7:00-8:00 hours for a movie, 8:00-8:30 hours for news, and so on. Again CNN runs its programmes during 0:00-3:00 hours for a movie, 3:00-4:00 hours for a serial, and so on. Similarly, other channels are engaged with programmes with definite slots. For graphical representation we consider each slot or programme as an interval on a real line. The interval can be represented as closed interval $[s_i, f_i]$, where s_i and f_i represent respectively the starting and the finishing time of the programme.

Each programme slot of a particular channel can thus be represented as an interval on 0 to 24 hours time interval. All the programme slots of all channels can be represented as a collection of intervals on the line segment $[0,24]$. Each interval has an weight which is equal to the average number of viewers watching the corresponding programme. This set of intervals forms a weighted interval graph G . An *interval graph* is a graph whose vertices can be mapped

into unique intervals on the real line such that two vertices in the graph are adjacent if and only if their corresponding intervals intersect. An interval graph is called *weighted* if its vertices have weights. Now the maximum number of viewers is equal to the weight of the k -colourable subgraph of the interval graph G . So this problem can be modelled as an interval graph.

Interval graphs have been extensively studied and used as models for many real world problems. The interval graph is one of the most useful discrete mathematical structure for modelling problems arising in the real world. It has many applications in various fields like archaeology, molecular biology, genetics, psychology, computer scheduling, storage information retrieval, electrical circuit design, traffic planning, VLSI design, etc [15, 39]. Interval graphs have been studied from both the theoretical and algorithmic points of view.

Various algorithmic problems concerning graphs in general and the graph colouring problem have been solved over last few years [3, 6, 7, 8, 14, 24, 25, 26, 28, 29, 30, 37, 45, 47].

Two graph colouring problems considered in the literature are:

- (i) the problem of finding minimum number of colours to colour all the vertices of a graph G so that no two adjacent vertices have the same colour. This is known as minimum colouring problem or optimal colouring problem,
- (ii) given k colours, the problem of finding maximum weight k -colourable subgraph of G .

In this paper, a set of television programs are considered as a set of intervals on a real line and it is shown that these intervals form an interval graph. The proposed daily life problem then modeled by k -colouring problem on interval graph. An algorithm is designed to solve this just-in-time optimization problem using $O(kMn^2)$ time, where n and M represent the total number of programmes of all channels and the upper bound of the viewers of all programmes of all channels respectively. This a very suitable application of k -colouring problem on interval graph.

1.1 Survey of related works

For arbitrary graphs, above problems are NP-complete [22]. A great deal of research has been focussed to identify classes of graphs for which these problems are solvable in polynomial time. Chordal graphs [2, 4, 18, 32, 46], interval graphs [15, 16, 17, 33, 44], planar graphs [5], outer planar graphs [9], trees [38], etc. are such classes.

The maximum weight k -colourable subgraph problem in chordal graphs is polynomially solvable when k is fixed and NP-hard when k is not fixed [46]. The maximum-weight independent set and maximum-weight k -independent set problems are also studied on many graph classes like permutation graphs [40, 41], trapezoid graphs [20], circular-arc graphs [27], etc. Efficient algorithms are designed to solve the maximal independent set problems on trapezoid graphs [19] and permutations graphs [36]. A minor variation of minimum colouring problem, called mutual exclusion scheduling is recently studied for interval graphs [13], permutation graphs [21] and

comparability graphs [21].

A parallel algorithm has been presented by Naor *et al.* [32] to solve optimal colouring problem for chordal graphs. Assuming that all the maximum cliques are part of the input, this algorithm runs in $O(\log^2 n)$ time using $O(n^3)$ processors.

Olariu [33] has solved the optimal colouring problem for interval graphs in $O(n + m)$ time, where n and m are the number of vertices and edges respectively, using greedy heuristic technique. Pal and Bhattacharjee [35] have designed a parallel algorithm to solve optimal colouring problem on interval graphs which takes $O(n/P + \log n)$ time using P processors.

Recently, just-in-time schedule algorithms have been designed to solve flow-shop problem [43], two-machine flow shop problem [11], single machine flow shop [31], etc. Kovalyov *et al.* [23] have discussed about the theory of fixed interval scheduling.

Recently, Saha, Pal and Pal [42] have solved 1-colouring problem for an interval graph whose vertex weights are taken as interval numbers. The proposed algorithm takes $O(n)$ time, for an interval graph with n vertices. This algorithm has been applied to solve the problem that involves selecting different programme slots (for a single session) telecast on different television channels in a day so as to reach the maximum number of viewers. In this problem they have assumed that the number of viewers of the programme slots are interval numbers.

1.2 The result obtained

In this paper, we have designed an algorithm to solve maximum weight k -colouring (MWkC) problem on interval graphs which takes $O(kMn^2)$ time, where n is the number of vertices and M is the upper bound on the weights of vertices. A set of programmes that are mutually exclusive in respect of time scheduling is called a *session*. This algorithm is used to select optimal programme slots which run in k parallel sessions such that the total number of viewers of the selected programmes is maximum. In this paper, it is proved that such programme slots can be selected using $O(kMn^2)$ time, where n is the total number of programmes telecasting in all channels during 24 hours and M is the least upper bound of the viewers among all programmes.

An outline to solve the proposed problem is given below.

The original problem is stated clearly as Problem P1 in Section 2. It is explained that this problem can be modelled as an interval graph G and the solution of MWkC problem on G is the solution of P1. This problem is stated as Problem P2. To solve the problem P2, a network N is constructed. The conventional k -flow problem on N is stated as Problem P3. This network flow problem determines the minimum weight k -flow. The maximum weight k -flow problem is stated as P4 and this problem is equivalent to the problem P2. Unfortunately, no suitable algorithm is available to solve maximum weight k -flow problem on a network. Thus, the maximum weight k -flow problem on N is converted to a minimum weight k -flow problem on the network N^U by applying a suitable transformation. This transformed problem is defined as Problem P5.

During the conversion, it is proved that the problem P1 is equivalent to P2, P2 is equivalent to P4 and P4 is equivalent to P5. Finally, the solution of the problem P1 is obtained from the solution of the problem P5.

2 Modelling of the Problem

Interval graphs are useful and significant in the process of modelling many real life situations, specially involving time dependencies. In this problem, we represent a programme slot as an interval. These slots or schedules of the programmes of a channel are denoted as vertices. If there exist intersection of timings in between two or more channels; this intersection is regarded as an edge between the vertices. If finishing time of a programme is the starting time of another programme then we assume that these programmes are non-intersecting. It may be noted that any two programmes in a particular television channel are non-intersecting. Various programmes have certain number of viewers which can be determined by statistical survey. The number of viewers of each programme is considered the weight of the corresponding vertex of the interval graph.

A *colouring* of a graph is an assignment of colours to its vertices so that no two adjacent vertices have the same colour. The vertices of one colour form a *colour class*. Any two vertices of a colour class are not adjacent. A k -colouring of a graph G uses k colours. The *chromatic number* χ is defined as the minimum k for which G has a k -colouring. A graph G is k -colouring if $\chi \leq k$ and is k -chromatic if $\chi = k$. A k -colouring of a k -chromatic graph is an *optimal colouring*. The weight of a k -colourable subgraph (this subgraph is k -chromatic) is the sum of weights of all vertices of the subgraph. Maximum weight k -colouring (MWkC) problem is to find a k -chromatic subgraph whose weight is maximum among all other k -chromatic subgraphs.

If a graph is k -chromatic then its vertex set V can be partitioned into k disjoint sets. Let $V = \{H_1, H_2, \dots, H_k\}$ be such a partition, where $H_i \cap H_j = \phi$, for all $i, j = 1, 2, \dots, k$, $i \neq j$, and for all $u, v \in H_i$, $i = 1, 2, \dots, k$, $(u, v) \notin E$ but if $u \in H_i$ and $v \in H_j$, there may be an edge between u and v . Thus, the colour i can be assigned to the set H_i . These sets H_i , $i = 1, 2, \dots, k$ are called chromatic partitions of V .

We first assume that AC wishes to run his advertisement from 0 to 24 hours in one session, that is, in a single duration the advertisement is to be shown in only one channel. This is equivalent to the maximum weight colouring problem on interval graph. The problem is also known as maximum weight 1-colouring problem (MW1C). But, if the company is interested to run the advertisement simultaneously in two parallel sessions it becomes maximum weight 2-colouring problem (MW2C). Hence for k parallel sessions it is termed as maximum weight k -colouring problem (MWkC). Now our problem is to determine the maximum weight subgraph which can be coloured using exactly k colours. The formal definition of the problem is given below.

Problem P1: Suppose a company or any organization is interested to run its advertisement simultaneously in k parallel sessions by selecting some programme slots. The restriction is that, any two programmes in the same session are disjoint. The objective of the problem is to select some programme slots such that the sum of the viewers of the selected programmes is maximum.

It is easy to observed that all the programme slots of all channels in a geographical area can be represented as a circular arc graph, which is a super class of interval graph. Here we assumed that at some point of time in a day (say, at 0:00 hours at midnight), all programmes that broadcasted earlier must terminate at or before 0:00 hours and all programmers in all channels would start broadcasting exactly at or after 0:00 hours and no programme would start earlier to 0:00 hours and continue after 0:00 hours.

So the **Problem P1** can be solved by solving the following Problem P2 on the constructed interval graph.

Problem P2: Find a subgraph $H(G)$ of an interval graph G which can be coloured using exactly k colours such that $\sum_{u \in H(G)} w(u)$ is maximum among all other such subgraphs, where $w(u)$ is the weight of the vertex u .

This problem is referred to as the maximum weight k -coloring problem. It is easy to observe that the Problems P1 and P2 are equivalent.

In the following sections, the solution procedure of the problem MWkC is discussed.

3 Cliques of Interval Graph

A *clique* of a graph is a set of vertices such that every two vertices in the set are joined by an edge. An *independent set* of a graph is the set of vertices such that any two vertices in the set are not connected by an edge. In a colouring, each colour class is an independent set, so G is k -colourable if and only if V is the union of k independent sets. Thus, ‘ k -colourable’ and ‘ k -partite’ (a graph is k -partite if its vertices can be expressed as the union of k independent sets) have the same meaning. The usage of the two terms are slightly different. The ‘ k -partite’ is a structural hypothesis, while ‘ k -colourable’ is the result of an optimization problem.

Let $G = (V, E)$, $V = \{1, 2, \dots, n\}$ be an interval graph and $\mathcal{C} = \{C_1, C_2, \dots, C_r\}$, for some r , be all maximal cliques. The maximal cliques of an interval graph satisfy the following result.

Lemma 1 *The maximal cliques of an interval graph G can be linearly ordered such that, for every vertex $u \in G$, the maximal cliques containing u occur consecutively [15].*

In an interval graph the *leading point* of a clique is the leftmost left endpoint of the interval at which all the other intervals in that clique intersect. It is assumed that the cliques are in order of their increasing leading point. It has been shown in [15] that for perfect graph the clique

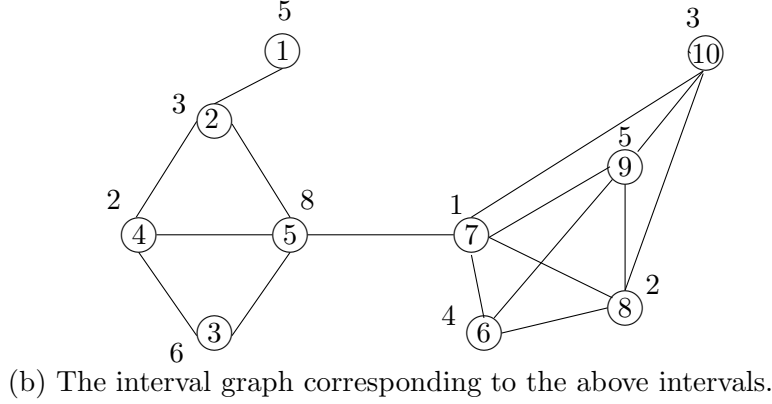
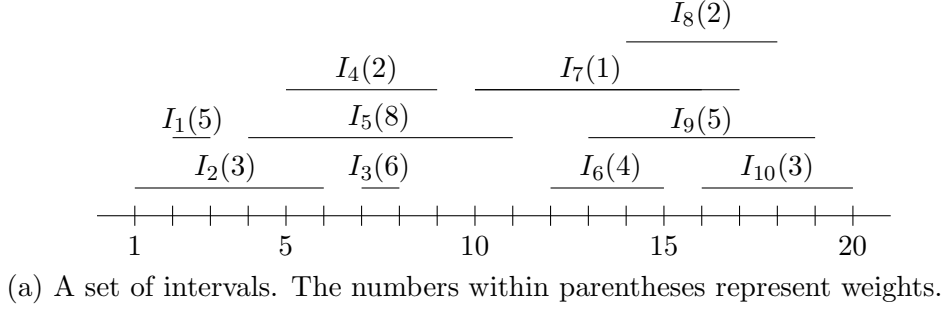


Figure 1: A set of intervals and its corresponding interval graphs.

number α (the size of a maximum clique) is equal to the chromatic number χ . Therefore, the vertex set V can be partitioned into α disjoint sets such that each of them is an independent set, i.e., $V = \cup_{i=1}^{\alpha} V_i$, where each V_i is an independent set and $V_i \cap V_j = \emptyset, i \neq j$. Thus, if $k \geq \alpha$ then $|V|$ is the maximum number of vertices in k -independent set. All the vertices of an independent set can be coloured by a single colour. Since interval graphs are perfect, therefore MWkC may be found by locating a maximum weight subgraph among all subgraphs by k -colouring. We propose to find such a subgraph for the interval graph.

For the graph of Figure 1(b), the maximal cliques are $C_1 = \{1, 2\}$, $C_2 = \{2, 4, 5\}$, $C_3 = \{3, 4, 5\}$, $C_4 = \{5, 7\}$, $C_5 = \{6, 7, 8, 9\}$, $C_6 = \{7, 8, 9, 10\}$.

The MWkC problem is solved by converting the problem to an equivalent problem on a network (Directed Acyclic Graph, in short DAG). Then solving the problem on DAG, the solution of Problem P2 is obtained and hence the solution of Problem P1. In the following section, DAG is defined and a method is described to construct a DAG for an interval graph.

4 The Network Flow Problem

A *network* N is a finite set of nodes and a subset of the ordered pairs (u, v) , $u \neq v$, called the *arcs*. The network N has a special return arc (t, s) , where node s is called the *source* in N and

node t is called the *sink* in N . The set of all arcs of N , except (t, s) is denoted by E_N . Further, a positive real-valued capacity $c(u, v) > 0$ and a non-negative weight $w_N(u, v)$ are associated with each edge (u, v) . For simplicity, the capacity for each non-existing edge is assumed to be zero, i.e., $c(u, v) = 0$ if $(u, v) \notin E$. A flow on G is a real valued function f on E_N if it satisfies the following three conditions:

$$\begin{aligned} f(u, v) &= -f(v, u), \text{ for all } (u, v) \in E_N, \\ f(u, v) &\leq c(u, v), \text{ for all } (u, v) \in E_N, \\ \sum_v f(u, v) &= 0, \text{ for all } v \in V - \{s, t\}. \end{aligned}$$

For each arc (u, v) of E_N , $f(u, v)$ represents the amount of flow in the arc (u, v) .

For a network N the minimum weight k -flow problem is defined for a network N as follows:

Problem P3: *The minimum weight k -flow problem is to obtain k edge disjoint paths P_1, P_2, \dots, P_k from the set of all possible paths from s to t in N to*

$$\text{minimize } \sum_{(u,v) \in J_k} w_N(u, v) f(u, v)$$

where

- (i) $f(u, v) = 0$ or 1 , for $(u, v) \in E_N$,
- (ii) $J_k = \cup_{i=1}^k E_N^i$,
- (iii) E_N^i is the set of arcs associated with the path P_i .

For each arc (u, v) of E_N , $f(u, v)$ represents the amount of flow in the arc (u, v) , and also it represents the net amount of flow from v to u in the rest of the network " $N - (u, v)$ ".

5 Construction of the Network

Let us consider a network N with nodes $C_0 (= s), C_1, C_2, \dots, C_r (= t)$ and arcs (C_{i-1}, C_i) , $i = 1, 2, \dots, r$, where C_0 is empty. Let each of these arcs, called a c -arc, be given a weight 0 and a capacity k .

By Lemma 1 for each $u \in V$ there exist consecutive cliques C_p, C_{p+1}, \dots, C_q , such that $u \in C_p, u \in C_{p+1}, \dots, u \in C_q$, $p \leq q$ but, $u \notin C_{p-1}, u \notin C_{q+1}$. Here we note that for $u \in V$ we have $u \in C_p, u \in C_q$ and $p \leq q$ but $u \notin C_i$ for any $i < p$ and $u \notin C_j$ for any $j > q$. We add an arc (C_{p-1}, C_q) to the network N and assign the weight $w(u)$ and capacity 1 to this arc. For each vertex $u \in V$, we get such an arc. Let these arcs be called the i -arcs. Let V_N and E_N be the set of nodes and the set of arcs of the network N respectively. Obviously, the network N is acyclic. Here the capacity of each i -arc is 1 and that of each c -arc is k . So, this network is referred also as integral flow network. We may replace each c -arc by k parallel arcs and assigning capacity 1 and weight 0 to each of them.

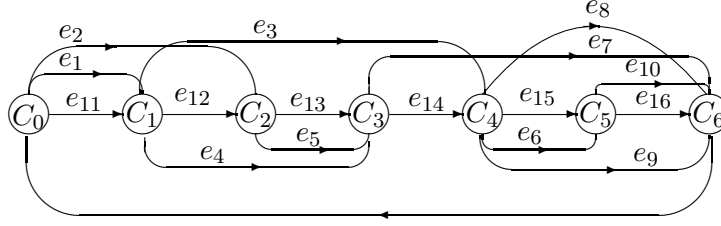


Figure 2: The network N for the graph of Figure 1(b). The (weight, capacity) of each arc are: $e_1 : (5, 1)$, $e_2 : (3, 1)$, $e_3 : (8, 1)$, $e_4 : (2, 1)$, $e_5 : (6, 1)$, $e_6 : (4, 1)$, $e_7 : (1, 1)$, $e_8 : (2, 1)$, $e_9 : (5, 1)$, $e_{10} : (3, 1)$, $e_{11} : (0, k)$, $e_{12} : (0, k)$, $e_{13} : (0, k)$, $e_{14} : (0, k)$, $e_{15} : (0, k)$, $e_{16} : (0, k)$.

The network N for the graph of Figure 1(b) is shown in Figure 2. The network has seven vertices $C_0, C_1, C_2, \dots, C_6$ and sixteen edges e_1, e_2, \dots, e_{16} of which e_1, e_2, \dots, e_{10} are i -arcs and $e_{11}, e_{12}, \dots, e_{16}$ are c -arcs. The (vertex, weight, capacity) of each i -arc of the network of Figure 2 is shown below:

$e_1 : (1, 5, 1)$, $e_2 : (2, 3, 1)$, $e_3 : (5, 8, 1)$, $e_4 : (4, 2, 1)$, $e_5 : (3, 6, 1)$, $e_6 : (6, 4, 1)$, $e_7 : (7, 1, 1)$, $e_8 : (8, 2, 1)$, $e_9 : (9, 5, 1)$, $e_{10} : (10, 3, 1)$.

Let us consider the maximum k -flow problem **P4** on the network N .

Problem P4: The maximum weight k -flow problem is to find the set of k disjoint paths P_1, P_2, \dots, P_k from the set of all possible paths from s to t in N to

$$\text{maximize} \quad \sum_{(u,v) \in J_k} w_N(u,v) f(u,v)$$

where

- (i) $J_k = \cup_{i=1}^k E_N^i$,
- (ii) E_N^i is the set of arcs associated with the path P_i ,
- (iii) the value of $f(u,v)$ is either 0 or 1 for all $(u,v) \in E_N$.

6 Properties of the Network N

The following result for a triangulated graph, given by Fulkerson and Gross [12], is also valid for an interval graph, because an interval graph satisfies the properties of triangulated graphs.

Lemma 2 A triangulated graph and so an interval graph with n vertices has at most n maximal cliques. The number of maximal cliques is n if and only if the graph has no edges [12].

The nodes of the network N are C_0, \dots, C_r , i.e., total number of nodes is $r + 1$. The arcs are of two types c -arc and i -arc. Each vertex is associated with an i -arc and each c -arc is drawn to

join two consecutive nodes of the network. Thus the total number of i -arc is n and that of c -arc is r . Hence, we can conclude the following result.

Lemma 3 *The total number of nodes and arcs of N are respectively $r + 1$ and $n + r + 1$ which are of $O(n)$, where $r < n$.*

The following lemma establishes the relationship between the arcs of the network N and the vertices of the interval graph G .

Lemma 4 *If (C_i, C_j) and (C_j, C_l) be two consecutive i -arcs of the set E_N then the corresponding vertices are non-adjacent in G .*

Proof. Let x and y be the vertices corresponding to the i -arcs (C_i, C_j) and (C_j, C_l) respectively. From the definition of N , it is clear that x belongs to the cliques $C_{i+1}, C_{i+2}, \dots, C_j$, but not in C_{j+1} and C_i and similarly, y belongs to the cliques $C_{j+1}, C_{j+2}, \dots, C_l$, but not in C_j and C_{l+1} . That is, there is no common clique for x and y . Hence, $(x, y) \notin E$. \square

The following lemma gives the guarantee about the existence of a k -flow in the network N .

Lemma 5 *The network N has a k -flow.*

Proof. Let $B_1 = \{C_0, C_1, \dots, C_i\}$ and $B_2 = \{C_j, C_{j+1}, \dots, C_r\}$ be two sets of vertices, for a given i , $1 \leq i \leq r$, where $j = i + 1$.

Case 1: Let there be no i -arcs between the nodes of B_1 and B_2 .

In this case every flow passes through the nodes $C_i \in B_1$ and $C_j \in B_2$ and the only connected arc is a c -arc with capacity k . Then there are at most k flows from s to t through C_i, C_j , because the flow of each i -arc of N is either 0 or 1.

Case 2: Let there be at least one i -arc between the nodes of B_1 and B_2 .

In this case, all flows do not necessarily pass through C_i and C_j . That is, there is at least one flow from a vertex of B_1 to a vertex of B_2 except C_i and C_j and k flows from C_i to C_j . Therefore, there is at least $k + 1$ flows from s to t . \square

The following theorem proves that the problems P2 and P4 are equivalent.

Theorem 1 *The problem P2 for G and problem P4 for N are equivalent.*

Proof. The construction of the network N from the weighted graph G shows that there is one to one correspondence between the set of vertices of G and the set of i -arcs of N . Also each i -arc corresponds to a vertex of G and the vertices corresponding to the i -arcs at the end of any c -arc are not directly connected. Hence each path from s to t in N corresponds to an independent set of G and conversely, each maximal independent set of G corresponds to a path from s to t in N . Further, we note that the weight of any c -arc is 0 and the weight of any i -arc is same as

the weight of the corresponding vertex of G . Thus, the total weight of any k colour classes of G and the total weight of the arcs of k paths of N are same. We take $f(x, y) = 1$ for each arc associated with these k paths and $f(x, y) = 0$ otherwise. Let the sets H_1, H_2, \dots, H_k of vertices of Problem P2 correspond to the set of paths P_1, P_2, \dots, P_k of Problem P4 respectively. Let $I_k = \cup_{i=1}^k H_i$.

Therefore,

$$\begin{aligned}
\sum_{v \in I_k} w(v) &= \sum_{i=1}^k \sum_{v \in H_i} w(v) \\
&= \sum_{i=1}^k \sum_{(x,y) \in E_N^i} w_N(x, y) \\
&\quad (\text{as } w_N(x, y) = w(v) \text{ for } i\text{-arc and} \\
&\quad w_N(x, y) = 0 \text{ for } c\text{-arc}) \\
&= \sum_{i=1}^k \sum_{(x,y) \in E_N^i} w_N(x, y) f(x, y) \\
&\quad (\text{since for all } i, f(x, y) = 1, \text{ for} \\
&\quad (x, y) \in E_N^i, \text{ and } f(x, y) = 0, \text{ otherwise}) \\
&= \sum_{(x,y) \in J_k} w_N(x, y) f(x, y).
\end{aligned}$$

Hence problems P2 and P4 are equivalent. \square

Unfortunately, no algorithm is available to solve the maximum weight k -flow problem. But, the minimum weight k -flow problem for N can be solved using the algorithm of Edmonds and Karp [10]. Thus we have to modify N by negating the weight of each arc and then finding a minimum weight k -flow. Unfortunately, the algorithm of Edmonds and Karp also requires that all arc weights are non-negative.

In order to convert a maximum weight flow problem to a minimum weight flow problem with positive arc weight a transformation is required.

To transform the problem, the array π is defined as follows:

$\pi(v_i) =$ largest weight of the path from v_i to t in N , $v_i \in V_N, i = 0, 1, 2, \dots, r$. The array π is computed by the Algorithm II.

ALGORITHM II

Input: The network N .

Output: The array $\pi(v_i)$, $v_i \in V_N$.

Initialization: $\pi(v_i) = 0$, $i = 1, 2, \dots, r$.

for $i = r - 1$ **to** 0 **step** -1 **do**

for each arc $(v_i, v_j) \in E_N$ **do**

```


$$\pi(v_i) = \max\{\pi(v_i), w_N(v_i, v_j) + \pi(v_j)\};$$

endfor;
endfor;
end  $\Pi$ 

```

The time complexity to calculate the array π is presented in the following lemma.

Lemma 6 *The array π can be computed correctly in $O(n)$ time.*

Proof. Let m_i be the total number of arcs adjacent to v_i . From Algorithm Π it follows that the time complexity of this algorithm is $\sum_{i=1}^{r-1} m_i = \text{total number of arcs of } N$, which is equal to $O(n)$ (from Lemma 3). As the network N is acyclic therefore, the correctness follows from the algorithm directly. Hence, the lemma follows. \square

The array π for the network N of Figure 2 is $\pi(C_0) = 20, \pi(C_1) = 15, \pi(C_2) = 13, \pi(C_3) = 7, \pi(C_4) = 7, \pi(C_5) = 3, \pi(C_6) = 0$.

7 Construction of a Minimum Weight Flow Network

Now, we convert the problem **P4** to a minimum weight k -flow problem **P5** as follows:

A network N^U is constructed from N using the same set of arcs (E_N), the same set of nodes (V_N) and identical capacities, but different weights. The weight on the arc (v_i, v_j) , $i < j$, is

$$w_{N^U}(v_i, v_j) = \pi(v_i) - \pi(v_j) - w_N(v_i, v_j),$$

for all $(v_i, v_j) \in E_N$.

The problem **P5** is defined as follows:

Problem P5: *For the network N^U find the set of k disjoint paths P_1, P_2, \dots, P_k from the set of all possible paths from s to t in N^U to*

$$\text{minimize} \quad \sum_{(u,v) \in J_k} w_{N^U}(u, v) f(u, v)$$

where

- (i) $J_k = \cup_{i=1}^k E_N^i$,
- (ii) E_N^i is the set of arcs associated with the path P_i ,
- (iii) the value of $f(u, v)$ is either 0 or 1 for all $(u, v) \in E_N$.

The Table 1 shows the weights of each arc of the networks N and N^U .

The following lemma establishes that the weight of each arc of the network N^U are non-negative.

Lemma 7 *The weights $w_{N^U}(v_i, v_j), i < j$ of all arcs of the network N^U are non-negative.*

arcs	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}
weights in N	5	3	8	2	6	1	2	5	3	0	0	0	0	0	0	0
weights in N^U	0	4	0	5	0	0	6	5	2	0	5	2	6	0	4	3

Table 1: Weights of arcs of the networks N and N^U .

Proof. If possible let the weight $w_{N^U}(v_i, v_j)$ of the arc (v_i, v_j) be negative. From the definition of N it follows that, in the left to right ordering of the cliques, the clique corresponding to v_i lies to the left of the clique corresponding to $v_j, i < j$. Since, $w_{N^U}(v_i, v_j) < 0$, therefore

$$\pi(v_i) - \pi(v_j) - w_N(v_i, v_j) < 0, \text{ or } \pi(v_i) < \pi(v_j) + w_N(v_i, v_j).$$

But, $\pi(v_i)$ is the largest weight of a path from v_i to t in N and similar is the interpretation for $\pi(v_j)$, so

$$\pi(v_i) \geq \pi(v_j) + w_N(v_i, v_j), i < j.$$

This contradicts the statement made earlier. Hence, $w_{N^U}(v_i, v_j) \geq 0$, for all $(v_i, v_j) \in E_N$. \square

From the definition of π it is clear that $\pi(s)$ is the largest weight of a path from s to t and $\pi(t) = 0$. Thus, for any arc $(v_i, v_j) \in E_N$, the upper bound of $w_{N^U}(v_i, v_j)$ is $\pi(s)$, which is proved in the following lemma.

Lemma 8 *The maximum value of weights of all arcs of the network N^U is $\pi(s)$, i.e., $\max_{i,j} w_{N^U}(v_i, v_j) = \pi(s)$.*

Proof. For $i \leq j$,

$$\begin{aligned} \max_{(v_i, v_j) \in E_N} w_{N^U}(v_i, v_j) &= \max_{(v_i, v_j) \in E_N} \{\pi(v_i) - [\pi(v_j) + w_N(v_i, v_j)]\} \\ &< \max_{(v_i, v_j) \in E_N} \pi(v_i) = \pi(s), \end{aligned}$$

since $\pi(v_j) + w_N(v_i, v_j) \geq 0$. Thus, the upper bound of the weights of any arc of the set E_N is the largest weight of a path from s to t in N . Hence, the lemma. \square

It can be shown that the maximum flow of N is equivalent to minimum flow of N^U .

Lemma 9 *For the same set of arcs the maximum weight flow from s to t in N is equal to the minimum weight flow in N^U .*

Proof. Without loss of generality, we assume that the sequence of arcs of a flow from s to t is $E'_N = \{(v_0, v_i), (v_i, v_j), (v_j, v_l), (v_l, v_r)\}$. Let Z_N and Z_{N^U} be the weights corresponding to this flow in the network N and N^U respectively.

Therefore,

$$\begin{aligned}
Z_{NU} &= \sum_{(v_i, v_j) \in E'_N} w_{NU}(v_i, v_j) \\
&= \sum_{(v_i, v_j) \in E'_N} \{\pi(v_i) - \pi(v_j) - w_N(v_i, v_j)\} \\
&= \pi(s) - \pi(t) - \sum_{(v_i, v_j) \in E'_N} w_N(v_i, v_j) \\
&= \pi(s) - \sum_{(v_i, v_j) \in E'_N} w_N(v_i, v_j) \quad (\text{as } \pi(t) = 0) \\
&= \pi(s) - Z_N \\
\text{i.e., } Z_N + Z_{NU} &= \pi(s),
\end{aligned}$$

which is constant, that is, independent of any flow. Therefore, if Z_{NU} is minimum then Z_N is maximum. Hence the lemma follows. \square

Let O_{P5} be the solution of the problem P5. That is, O_{P5} is a set of k disjoint paths and each path is a sequence of vertices and edges (i -arcs and c -arcs). Again, let O_{P4} be the solution of the problem P4 and it is also the set of k disjoint paths containing the same sets of vertices and edges of O_{P5} . Note that k disjoint paths of P4 and P5 are same, but the arc weights are different.

Let O'_{P4} and O'_{P5} be the sets of i -arcs of O_{P4} and O_{P5} respectively. Now from the above theorem it follows that O'_{P5} is equal to O'_{P4} .

Combining Lemma 9 and Theorem 1 the following result can be stated.

Theorem 2 *The vertices of G corresponding to the arcs of O'_{P5} are the vertices of the problem P2 for the interval graph G .*

In the next section, the algorithm to solve MWkC problem is presented. Also, the time and space complexities are analyzed.

8 The Algorithm and its Complexity

We have discussed several properties about network N in previous sections. Also, we have shown that maximum weight k -flow problem is equivalent to MWkC problem. Again, the solution of MWkC is the solution of the problem P1. In the following, we present the major steps of the proposed algorithm to solve maximum weight k -flow problem.

ALGORITHM MWKF

Input: The sorted endpoints list of intervals of an interval graph G and a positive integer k .

Output: The k colour classes H_1, H_2, \dots, H_k of the graph G .

Step 1: Find all maximal cliques of the graph G .

Step 2: Construct a network N , using the technique described in Section 5.

Step 3: Compute the array π using Algorithm II.

Step 4: Convert the network N to the network N^U by changing the weights to
 $w_{N^U}(v_i, v_j) = \pi(v_i) - \pi(v_j) - w_N(v_i, v_j)$, for all $(v_i, v_j) \in E_N$.

Step 5: Solve the minimum weight k -flow problem for the network N^U .

Step 6: For any i -arc $(u, v) \in E_N$, if $f(u, v) > 0$ then put the corresponding vertex to the set Q .

Step 7: Distribute the vertices of Q to k colour classes H_1, H_2, \dots, H_k . The vertices of each set have same colour.

end MWKF

In Figure 3, the dotted arcs are the arcs of the minimum weight 2-flow for the network N^U .

The arcs of the minimum weight 2-flow of the network N^U are $e_1, e_2, e_3, e_5, e_6, e_9, e_{10}$ and $\{1, 2, 5, 3, 6, 9, 10\}$ are the vertices corresponding to the arcs $e_1, e_2, e_3, e_5, e_6, e_9, e_{10}$. The set Q after Step 6 of Algorithm MWKF is $\{1, 2, 3, 5, 6, 9, 10\}$. The total weight (maximum) of 2-flow is 34 (the sum of weights of the vertices of Q) and the 2 flows are $\{1, 5, 6, 10\}$ and $\{2, 3, 9\}$. The other solutions are $\{1, 5, 9\}$, $\{2, 3, 6, 10\}$.

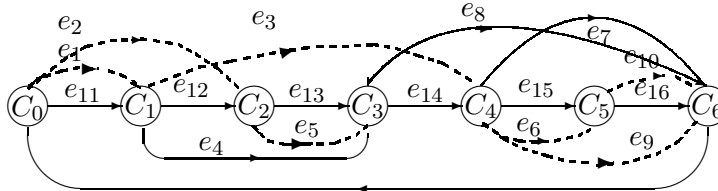


Figure 3: Dotted arcs are the solution of the MW2C problem of the graph of Figure 1.

Theorem 3 *The running time of Algorithm MWKF is $O(kn\sqrt{\log c} + m)$, where n and m represent respectively the number of vertices and edges and c is the weight of the longest path of the interval graph.*

Proof. All maximal cliques of an interval graph can be computed in $O(n + \gamma)$ time, where γ is the total size of all maximal cliques [34]. But, γ is of order $O(n + m)$ [15]. The network N can be constructed using $O(n)$ time. To compute the array π and the weight of each arc of the network N^U take $O(n)$ time. The minimum weight k -flow problem of N^U can be solved using the algorithm of Edmonds and Karp in time $O(k \times (\text{complexity of shortest path problem}))$. An $O(m + n\sqrt{\log c})$ time algorithm [1] is available to solve the shortest path problem on general

graph with n vertices and m edges, where cost of each arc is non-negative integer number bounded by c . In N^U , the weight of each arc is bounded by $\pi(s)$ ($\leq c$), if c is the largest weight of a path in the given interval graph. Thus, since there are $O(n)$ arcs in N^U , the algorithm requires $O(kn\sqrt{\log c} + m)$ operations to solve the k -flow problem. The Step 6 requires only $O(n)$ time. The Step 7 can be computed using $O(kn)$ time. Therefore, the total time complexity is $O(kn\sqrt{\log c} + m)$. \square

The upper bound of the weights of arcs in N or N^U is $\pi(s)$. If M be the upper bound of weights of the vertices of the given interval graph then the upper bound of c is nM . From this analogy we can draw the following conclusion.

Theorem 4 *The running time of Algorithm MWKF is $O(kMn^2)$, where n and M represent respectively the number of vertices and the upper bound of the weights of the vertices of the interval graph.*

From this theorem one can conclude that the maximum weight k -colourable subgraph problem on interval graph can be solved using $O(kMn^2)$ time, where n and M represent respectively the number of vertices and the upper bound of weights of the vertices of the interval graph.

At the beginning of this article, it is mentioned that a daily life problem is considered in this paper. This practical problem is solved by using the concept of graph theory. In the next section, a numerical example is considered for illustration.

9 Numerical Illustration

We shall use some standard notations to name the television programme for convenience of solving the problem. The name of the programmes of various channels and the number of viewers (taken as the weight of the interval) are written in parentheses. The first number of the parentheses represents the name of the programme and the second number tells about weight or strength of viewer in lakh.

Here we consider three channels and try to find out a solution of the problem. The channels we consider are National Geographic, Discovery, and AXN. Some programmes of National Geographic are as follows: 8:00–9:00 a.m. India Diaries: Keeping faith ($N_1, 5$); 9:00–10:00 a.m. Reel people : Through these eyes ($N_2, 7$); 10:00–10:30 a.m. Mission Everest ($N_3, 8$); 10:30–11:00 a.m. Nick's Quest ($N_4, 3$); 11:00–11:30 p.m. Wild orphan ($N_5, 4$); 11:30–12:00 noon Myths and logic of shoolin kung fu ($N_6, 5$); 12:00–1:00 p.m. Adventure atarts here with Toyota ($N_7, 4$) and so on.

The programmes of Discovery channel are as follows:
8:00–9:00 a.m. Terra X ($D_1, 5$); 9:00–10:00 a.m. Tunk yard war kids ($D_2, 7$); 10:00–10:30 a.m. Discover India ($D_3, 8$); 10:30–11:00 a.m. Real kids real adventure ($D_4, 3$); 11:00–11:30 a.m.

Time	8:00-9:00	9:00-10:00	10:00-10:30	10:30-11:00	11:00-11:30	11:30-12:00	12:00-13:00
Short name	N_1	N_2	N_3	N_4	N_5	N_6	N_7
Viewers	5	7	8	3	4	5	4
Time	13:00-13:30	13:30-14:00	14:00-14:30	14:30-15:30	15:30-16:00	16:00-17:00	17:00-17:30
Short name	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}
Viewers	7	8	2	4	5	6	4
Time	17:30-18:00	18:00-18:30	18:30-19:00	19:00-19:30	19:30-20:30	20:30-21:30	21:30-22:00
Short name	N_{15}	N_{16}	N_{17}	N_{18}	N_{19}	N_{20}	N_{21}
Viewers	3	1	2	3	6	1	4
Time	22:00-22:30	22:30-23:00					
Short name	N_{22}	N_{23}					
Viewers	3	1					

Table 2: Programmes of the channel National Geography.

Time	8:00-9:00	9:00-10:00	10:00-10:30	10:30-11:00	11:00-11:30	11:30-12:00	12:00-12:30
Short name	D_1	D_2	D_3	D_4	D_5	D_6	D_7
Viewers	5	7	8	3	4	5	4
Time	12:30-13:00	13:00-14:00	14:00-15:00	15:00-16:00	16:00-17:00	17:00-18:00	18:00-18:30
Short name	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}
Viewers	7	8	2	4	5	6	4
Time	18:30-19:00	19:00-20:00	20:00-21:00	21:00-22:00	22:00-23:00		
Short name	D_{15}	D_{16}	D_{17}	D_{18}	D_{19}		
Viewers	2	3	1	4	2		

Table 3: Programmes of the channel Discovery.

Eccentriks (D_5 , 4); 11:30–12:00 noon Djuma : South Africa (D_6 , 5); 12:00–12:30 p.m. Wedding Story (D_7 , 4), etc.

The programmes of AXN with time and number of viewers are such as 7:00–8:00 a.m. Relic Hunter (A_1 ,3); 8:00–9:00 a.m. Now see this (A_2 ,1); 9:00–10:00 a.m. Guinness World records (A_3 ,2); 10:00–11:00 a.m. Ripley’s Believe it or Not (A_4 ,3), etc.

The tabular representation of the programmes of different channels are shown in the tables 2, 3 and 4.

The combined programme slots of three channels are shown in the Figure 4.

Time	7:00-8:00	8:00-9:00	9 :00-10:00	10:00-11:00	11:00-12:00	12:00-15:00	15:00-17:30
Short name	A_1	A_2	A_3	A_4	A_5	A_6	A_7
Viewers	3	1	2	3	2	5	4
Time	17:30-18:00	18:00-19:00	19:00-20:00	20:00-22:30	22:30-24:00		
Short name	A_8	A_9	A_{10}	A_{11}	A_{12}		
Viewers	3	4	2	4	2		

Table 4: Programmes of the channel AXN.

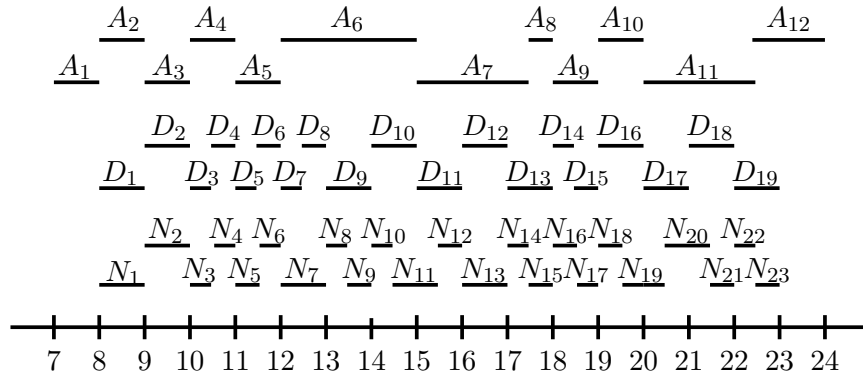


Figure 4: The set of intervals for the three channels. The programme slots are shown in the tables 2, 3, 4.

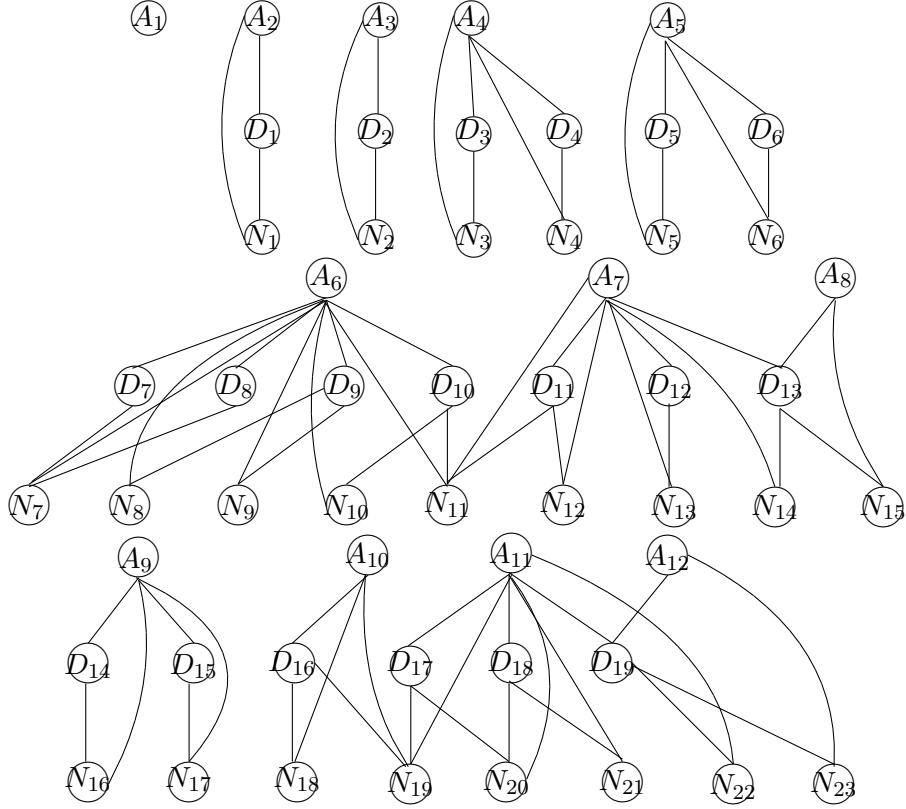


Figure 5: The interval graph corresponding to the intervals of Figure 4.

The corresponding interval graph is shown in Figure 5.

For the above example it is observed that the interval graph is disconnected and has eight components. But, if the large number of channels are considered then the graph may be connected. If the graph becomes disconnected then apply the algorithm MWKF to each component and combine the solutions obtained from all components. The combined solution is the final solution of the problem.

9.1 Computational result

The algorithm is implemented in C and the results for the problem consider in this section are given below. The weights corresponding to the programmes are taken randomly between 1 and 9.

Step 1: All maximal cliques are

$$\begin{aligned}
 c_1 &= (1), c_2 = (2, 3, 4), c_3 = (5, 6, 7), c_4 = (8, 9, 10), c_5 = (10, 11, 12), c_6 = (13, 14, 15), \\
 c_7 &= (15, 16, 17), c_8 = (18, 20, 25), c_9 = (19, 20, 25), c_{10} = (21, 22, 25), c_{11} = (22, 23, 25), \\
 c_{12} &= (24, 25, 26), c_{13} = (25, 26, 27), c_{14} = (27, 29, 32), c_{15} = (28, 29, 32), \\
 c_{16} &= (30, 31, 32), c_{17} = (32, 33, 34), c_{18} = (34, 35, 36), c_{19} = (37, 38, 41),
 \end{aligned}$$

$$c_{20} = (39, 40, 41), c_{21} = (42, 43, 44), c_{22} = (43, 44, 45), c_{23} = (45, 46), \\ c_{24} = (46, 47, 50), c_{25} = (47, 48, 50), c_{26} = (48, 49, 50), c_{27} = (50, 51, 52), c_{28} = (52, 53, 54).$$

Step 2: The nodes of N are

$$C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, \\ C_{15}, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, C_{22}, C_{23}, C_{24}, C_{25}, C_{26}, C_{27}, C_{28}.$$

$$i\text{-arcs: } e_1 : (C_0, C_1), e_2 : (C_1, C_2), e_3 : (C_1, C_2), e_4 : (C_1, C_2), e_5 : (C_2, C_3), e_6 : (C_2, C_3), \\ e_7 : (C_2, C_3), e_8 : (C_3, C_4), e_9 : (C_3, C_4), e_{10} : (C_3, C_5), e_{11} : (C_4, C_5), e_{12} : (C_4, C_5), \\ e_{13} : (C_5, C_6), e_{14} : (C_5, C_6), e_{15} : (C_5, C_7), e_{16} : (C_6, C_7), e_{17} : (C_6, C_7), e_{18} : (C_7, C_8), \\ e_{19} : (C_8, C_9), e_{20} : (C_7, C_9), e_{21} : (C_9, C_{10}), e_{22} : (C_9, C_{11}), e_{23} : (C_{10}, C_{11}), e_{24} : (C_{11}, C_{12}), \\ e_{25} : (C_7, C_{13}), e_{26} : (C_{11}, C_{13}), e_{27} : (C_{12}, C_{14}), e_{28} : (C_{14}, C_{15}), e_{29} : (C_{13}, C_{15}), e_{30} : (C_{15}, C_{16}), \\ e_{31} : (C_{15}, C_{16}), e_{32} : (C_{13}, C_{17}), e_{33} : (C_{16}, C_{17}), e_{34} : (C_{16}, C_{18}), e_{35} : (C_{17}, C_{18}), e_{36} : (C_{17}, C_{18}), \\ e_{37} : (C_{18}, C_{19}), e_{38} : (C_{18}, C_{19}), e_{39} : (C_{19}, C_{20}), e_{40} : (C_{19}, C_{20}), e_{41} : (C_{18}, C_{20}), e_{42} : (C_{20}, C_{21}), \\ e_{43} : (C_{20}, C_{22}), e_{44} : (C_{20}, C_{22}), e_{45} : (C_{21}, C_{23}), e_{46} : (C_{22}, C_{24}), e_{47} : (C_{23}, C_{25}), e_{48} : (C_{24}, C_{26}), \\ e_{49} : (C_{25}, C_{26}), e_{50} : (C_{23}, C_{27}), e_{51} : (C_{26}, C_{27}), e_{52} : (C_{26}, C_{28}), e_{53} : (C_{27}, C_{28}), e_{54} : (C_{27}, C_{28}).$$

$$c\text{-arcs: } e_{54+i} : (C_{i-1}, C_i), i = 1, 2, \dots, 28.$$

Step 3: π values of the nodes:

$$\pi(C_0) = 110, \pi(C_1) = 107, \pi(C_2) = 102, \pi(C_3) = 95, \pi(C_4) = 87, \pi(C_5) = 84, \pi(C_6) = 80, \\ \pi(C_7) = 75, \pi(C_8) = 71, \pi(C_9) = 64, \pi(C_{10}) = 57, \pi(C_{11}) = 49, \pi(C_{12}) = 47, \pi(C_{13}) = 42, \\ \pi(C_{14}) = 43, \pi(C_{15}) = 38, \pi(C_{16}) = 32, \pi(C_{17}) = 28, \pi(C_{18}) = 25, \pi(C_{19}) = 21, \pi(C_{20}) = 19, \\ \pi(C_{21}) = 16, \pi(C_{22}) = 10, \pi(C_{23}) = 10, \pi(C_{24}) = 9, \pi(C_{25}) = 9, \pi(C_{26}) = 5, \pi(C_{27}) = 2, \\ \pi(C_{28}) = 0.$$

Step 4: The weights of the arcs of the network N^U .

$$i\text{-arcs: } e_1 = 0, e_2 = 4, e_3 = 0, e_4 = 0, e_5 = 5, e_6 = 0, e_7 = 0, e_8 = 0, e_9 = 0, e_{10} = 8, e_{11} = 0, \\ e_{12} = 0, e_{13} = 0, e_{14} = 0, e_{15} = 7, e_{16} = 0, e_{17} = 0, e_{18} = 0, e_{19} = 0, e_{20} = 7, e_{21} = 0, e_{22} = 7, \\ e_{23} = 0, e_{24} = 0, e_{25} = 28, e_{26} = 5, e_{27} = 0, e_{28} = 0, e_{29} = 0, e_{30} = 1, e_{31} = 0, e_{32} = 10, e_{33} = 0, \\ e_{34} = 1, e_{35} = 0, e_{36} = 0, e_{37} = 0, e_{38} = 3, e_{39} = 0, e_{40} = 0, e_{41} = 2, e_{42} = 0, e_{43} = 6, e_{44} = 7, \\ e_{45} = 0, e_{46} = 0, e_{47} = 0, e_{48} = 0, e_{49} = 0, e_{50} = 4, e_{51} = 0, e_{52} = 3, e_{53} = 1, e_{54} = 0.$$

$$c\text{-arcs: } e_{55} = 3, e_{56} = 5, e_{57} = 7, e_{58} = 8, e_{59} = 3, e_{60} = 4, e_{61} = 5, e_{62} = 4, e_{63} = 7, e_{64} = 7, \\ e_{65} = 8, e_{66} = 2, e_{67} = 5, e_{68} = 1, e_{69} = 5, e_{70} = 6, e_{71} = 4, e_{72} = 3, e_{73} = 4, e_{74} = 2, e_{75} = 3, \\ e_{76} = 6, e_{77} = 0, e_{78} = 1, e_{79} = 0, e_{80} = 4, e_{81} = 3, e_{82} = 2.$$

Step 5: Here we consider $k = 2$. The two paths P_1 and P_2 are given below.

$$P_1 : C_0 - e_1 - C_1 - e_3 - C_2 - e_6 - C_3 - e_8 - C_4 - e_{11} - C_5 - e_{13} - C_6 - e_{16} - C_7 - e_{18} - C_8 - e_{19} - \\ C_9 - e_{21} - C_{10} - e_{23} - C_{11} - e_{24} - C_{12} - e_{27} - C_{14} - e_{28} - C_{15} - e_{31} - C_{16} - e_{33} - C_{17} - e_{35} - \\ C_{18} - e_{37} - C_{19} - e_{40} - C_{20} - e_{42} - C_{21} - e_{45} - C_{23} - e_{47} - C_{25} - e_{49} - C_{26} - e_{51} - C_{27} - e_{52} - C_{28}.$$

$P_2 : C_0 - e_{55} - C_1 - e_4 - C_2 - e_7 - C_3 - e_9 - C_4 - e_{12} - C_5 - e_{14} - C_6 - e_{17} - C_7 - e_{20} - C_9 - e_{22} - C_{11} - e_{26} - C_{13} - e_{29} - C_{15} - e_{30} - C_{16} - e_{34} - C_{18} - e_{38} - C_{19} - e_{39} - C_{20} - e_{43} - C_{22} - e_{46} - C_{24} - e_{48} - C_{26} - e_{52} - C_{28}$.

Step 6 and Step 7: The set of i -arcs on the path P_1 is

$$X_1 = \{e_1, e_3, e_6, e_8, e_{11}, e_{13}, e_{16}, e_{18}, e_{19}, e_{21}, e_{23}, e_{24}, e_{27}, e_{28}, \\ e_{31}, e_{33}, e_{35}, e_{37}, e_{40}, e_{42}, e_{45}, e_{47}, e_{49}, e_{51}, e_{52}\}$$

and for the path P_2 is

$$X_1 = \{e_4, e_7, e_9, e_{12}, e_{14}, e_{17}, e_{20}, e_{22}, e_{26}, e_{29}, e_{30}, e_{34}, e_{38}, e_{39}, e_{43}, e_{46}, e_{48}, e_{52}\}.$$

The set of vertices corresponding to the i -arcs of X_1 and X_2 are

$$H_1 = \{A_1, N_1, N_2, N_3, N_4, N_5, N_6, D_7, D_8, N_8, N_9, N_{10}, N_{11}, N_{12}, N_{13}, N_{14}, N_{15}, D_{14}, \\ D_{15}, N_{18}, N_{19}, N_{20}, N_{21}, N_{22}, A_{12}\}$$

and $H_2 = \{D_1, D_2, D_3, D_4, D_5, D_6, N_7, D_9, D_{10}, D_{11}, D_{12}, D_{13}, N_{16}, N_{17}, D_{16}, D_{17}, D_{18}, D_{19}\}$.

The weights of H_1 and H_2 are respectively 110 and 74 and the total weight of 2-colour set is 184.

Another 2-colour set of this problem is given below.

$$H_1 = \{N_1, N_2, N_3, N_4, N_5, N_6, N_7, N_8, N_9, N_{10}, N_{11}, N_{12}, N_{13}, N_{14}, N_{15}, N_{16}, N_{17}, N_{18}, N_{19}, \\ N_{20}, N_{21}, N_{22}, A_{12}\}$$

$$H_2 = \{A_1, D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}, D_{11}, D_{12}, D_{13}, D_{14}, D_{15}, D_{16}, D_{17}, D_{18}, D_{19}\}.$$

In this solution the weights of H_1 and H_2 are respectively 97 and 87 and the total weight of 2-colour set is also 184.

Note that the weight of two 2-colour sets are same and this is equal to **184**. Also, in both the cases

- (i) $H_1 \cap H_2 = \phi$,
- (ii) $H_1 \cup H_2 \subset V$,
- (iii) Each colour set H_1 and H_2 forms an independent set,
- (iv) Weight of $H_1 \cup H_2$ is maximum among all other 2-colour sets.

The following result follows from Theorem 4.

Theorem 5 *The programme slots for k parallel sessions with maximum number of viewers can be selected using $O(kMn^2)$ time, where n is the total number of programmes in all channels and M is the upper bound of viewers of all programmes of all channels.*

10 Conclusion

In this paper, a method is described for the ACs to display their advertisement in k parallel sessions in different television channels. The objective of the ACs are to catch the maximum number of viewers. The companies want to attract more and more viewers through their advertisement. This real life problem is modelled as an interval graph. The various programme slots are considered as interval. The method we have developed has been solved by converting the

problem into maximum weight k -colouring problem on interval graph. The number of viewers of a particular programme slot is taken as the weight of that corresponding interval. While solving the problem we have not considered the subscription rate of the programmes of various channels. We are trying to solve this problem by taking into account the subscription rate as another objective.

It is obvious, that the companies try to select such programme slots whose number of viewers are very high. Sometimes, it may also happen that the subscription rate of a particular programme having large number of viewers is very high. It becomes difficult for a company to afford this high price. In this case, the company may discard this high price programme slot using our proposed algorithm by modifying the weight of the corresponding programme slot or interval by assigning the weight to zero.

References

- [1] Ahuja, R.K., Mehlhorn, K., Orlin, J.B., & Tarjan, R.E. (1990). Faster algorithms for the shortest path problem. *J. ACM*, 37, 213–223.
- [2] Balas, E., & Xue, J. (1991). Minimum weighted coloring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs. *SIAM J. Comput.*, 20, 209–221.
- [3] Blöchliger, I., & Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research*, 35 (3), 960–975.
- [4] Bodlaender, H., & Kloks, T. (1993). A simple linear time algorithm for triangulating three-colored graphs. *J. Algorithms*, 15, 160–172.
- [5] Boyar, J.F., & Karloff, H.J. (1987). Coloring planar graphs in parallel. *J. Algorithms*, 8, 470–479.
- [6] Bui, T.N., Nguyen, T.H., Patel, C.M., & Phan, K.-A.T. (2008). An ant-based algorithm for coloring graphs, *Discrete Applied Mathematics*, 156 (2), 190–200.
- [7] Caramiá, M., & DellOlmo, P. (2008). Coloring graphs by iterated local search traversing feasible and infeasible solutions. *Discrete Applied Mathematics*, 156 (2), 201–217.
- [8] Chiarandini, M., & Stützle, T. (2010) An analysis of heuristics for vertex colouring. In P. Festa (ed.), *Experimental Algorithms, Proceedings of the 9th International Symposium*. (SEA 2010), vol. 6049 of Lecture Notes in Computer Science, pp. 326–337.

- [9] Disk, K., Hagerup, T., & Rytter, W. (1989). Optimal parallel algorithms for the recognition and colouring outerplanar graphs. *LNCS, 379, Mathematical Foundations of Computer Science*, 207–217.
- [10] Edmonds, J., & Karp, R.M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19, 248–264.
- [11] Elalouf, A., Levner, E., & Tang, H. (2013). An improved FPTAS for maximizing the weighted number of just-in-time jobs in a two-machine flow shop problem. *Journal of Scheduling*, 16 (4), 429–435.
- [12] Fulkerson, D.R., & Gross, O.A., (1965). Incidence matrices and intervals graphs. *Pacific J. Math.* 15, 835–855.
- [13] Gardi, F. (2006). Mutual exclusion scheduling with interval graphs or related classes: complexity and algorithms. *4OR: A Quarterly J. of Operations Research*, 4(1), 87–90.
- [14] Galinier, P., Hertz, A., & Zufferey, N. (2008). An adaptive memory algorithm for the k -coloring problem. *Discrete Applied Mathematics*, 156 (2), 267–279.
- [15] Golumbic, M.C. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York.
- [16] Gupta, U.I., Lee, D.T., & Leung, J.Y.-T. (1982). Efficient algorithms for interval graphs and circular-arc graphs. *Network*, 12, 459–467.
- [17] Gyarfás, A., & Lehel, J. (1985). Covering and coloring problems for relatives of intervals. *Discrete Math.*, 55, 167–180.
- [18] Ho, C.W., & Lee, R.C.T. (1988). Efficient parallel algorithms for finding maximal cliques, clique trees and minimum coloring on chordal graphs. *Information Processing Letters*, 28, 301–309.
- [19] Hota, H., Pal, M., & Pal, T.K. (1999). An efficient algorithm to generate all maximal independent sets on trapezoid graphs. *Intern. J. Computer Mathematics*, 70, 587–599.
- [20] Hota, M., Pal, M., & Pal, T.K. (2001). An efficient algorithm for finding a maximum weight k -independent set on trapezoid graphs. *Computational Optimization and Applications*, 18, 49–62.
- [21] Jansen, K. (2003). The mutual exclusion scheduling problem for permutation and comparability graphs. *Information and Computation*, 180 (2), 71–81.
- [22] Karp, R.M. (1972). *Reducibility among combinatorial problems, in complexity of computer computations*. R.E.Miller and J.W.Thacher, editors, Plenum Press.

- [23] Kovalyov, M.Y., Ng, C.T., & Cheng, T.C.E. (2007). Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research*, 178 (2), 331-342.
- [24] Lewis, R. (2009). A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. *Computers & Operations Research*, 36 (7), 2295-2310.
- [25] Lü, Z., & Hao, J.-K. (2010). A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203 (1), 241-250.
- [26] Lucet, C., Mendes, F., & Moukrim, A. (2006). An exact method for graph coloring. *Computers & Operations Research*, 33 (8), 2189-2207.
- [27] Manacher, G.L., & Mankus, T.A. (2002). A simple linear time algorithm for finding a maximum independent set of circular arcs using intervals alone. *Networks*, 39(2), 68-72.
- [28] Malaguti, E., Monaci, M., & Toth, P. (2008). A Metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, 20 (2), 302-316.
- [29] Malaguti, E., Monaci, M., & Toth, P. (2010). An exact approach for the vertex coloring problem. *Discrete Optimization*, In Press.
- [30] Méndez-Díaz, I., & Zabala, P. (2006). A Branch-and-Cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154 (5), 826-847.
- [31] Mosheiov, G., & Shabtay, D. (2013). Maximizing the weighted number of just-in-time jobs on a single machine with position-dependent processing times. *Journal of Scheduling*, 16 (5), 519-527.
- [32] Naor, J., Naor, M., & Schaffer, A.A. (1989). Fast parallel algorithms for chordal graphs. *SIAM J. Comput.*, 18, 327-349.
- [33] Olariu, S. (1991). An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37, 21-25.
- [34] Pal, M., & Bhattacharjee, G.P. (1995). An optimal parallel algorithm for computing all maximal cliques of an interval graph and its applications. *J. of Institution of Engineers (India)*, 76, 29-33.
- [35] Pal, M., & Bhattacharjee, G.P. (1996). An optimal parallel algorithm to color an interval graph. *Parallel Processing Letters*, 6, 439-449.
- [36] Pal, M. (1998). A parallel algorithm to generate all maximal independent sets on permutation graphs. *Intern. J. Computer Mathematics*, 67, 261-274.

- [37] Plumettaz, M., Schindl, D., & Zufferey, N. (2010). Ant local search and its efficient adaptation to graph colouring. *Journal of Operational Research Society*, 61 (5), 819–826.
- [38] Rajeani, P. (1992). Optimal parallel 3-coloring algorithm for rooted tree and its applications. *Information Processing Letters*, 41, 153–156.
- [39] Roberts, F.S. (1978). *Graph Theory and its Application to Problems of Society*, SIAM, Philadelphia, PA.
- [40] Saha, A., Pal, M., & Pal, T.K., (2003). Maximum weight k -independent set problem on permutation graphs. *Intern. J. Computer Mathematics*, 80 (12), 1477–1487.
- [41] Saha, A., Pal, M., & Pal, T.K., (2005). An efficient PRAM algorithm for maximum-weight independent set on permutation graphs. *Journal of Applied Mathematics and Computing*, 19 (1+2), 77–92.
- [42] Saha, A., Pal, M., & Pal, T.K., (2007). Selection of programme slots of television channels for giving advertisement: a graph theoretic approach. *Information Sciences*, 177, 2480–2492.
- [43] Shabtay, D. (2012). The just-in-time scheduling problem in a flow-shop scheduling system. *European Journal of Operational Research*, 216 (3), 521–532.
- [44] Slusarek, M. (1989). A coloring algorithm for interval graphs. *LNCS, 379, Mathematical Foundations of Computer Science*, 471–480.
- [45] Talaván, P.M., & Yáñez, J. (2008). The graph coloring problem: A neuronal network approach. *European Journal of Operational Research*, 191 (1), 100–111.
- [46] Yannakakis, M., & Gavril, F. (1987). The maximum k -colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24, 133–137.
- [47] Zufferey, N., Amstutz, P., & Giaccari, P. (2008). Graph colouring approaches for a satellite range scheduling problem. *Journal of Scheduling*, 11 (4), 263–277.